VZBASCOM

by Bob Kitch

VZBASCOM

A BASIC-Compiler for the VZ-200 and VZ-300 computers.

VZBASCOM is an interactive BASIC Compiler. When loaded it supplements the BASIC Interpreter in memory. The SOURCE program is entered into the computer in BASIC, as with a normal BASIC program, or read in from disk or tape. After input is complete, the compiler is called on to produce an autorun machine code program, that the compiler can immediately execute, from the source program. In this way the speed of program execution is increased 20-200 times compared to the BASIC Interpreter.

You can jump back and forth between the SOURCE program (in BASIC) and the compiled OBJECT code (in Machine Language). This has the advantage, that corrections and changes to the compiled program can be made simply, and as often as desired. With appropriate memory extension (VZ-300 with 16K expansion, VZ-200 with 64K expansion) you can test run the OBJECT code, without having to reload the SOURCE code or the compiler afterwards, unless your program is so error-ridden that it wipes the memory area.

The minimum equipment necessary is a VZ-300 with a 16K expansion.

VZBASCOM uses a large number of the standard BASIC commands. Some, however, cannot be used, but there are a number of additional functions, that are not available in normal BASIC. Read through these instructions carefully, to become familiar with the available command set.

SOURCE program and OBJECT code can be saved either on diskette or cassette.

The only variables allowed are integer from +32767 to -32768 and strings. It follows, therefore, that VZBASCOM is not suitable for mathematical applications. If however, what you want is to create fast, action games and you only have the BASIC language to serve you, then VZBASCOM is the right partner.

Loading

VZBASCOM will load and run from disk :-

```
BRUN"VZBASCOM"
```

At the start an introductory screen displays:

```
***** V Z B A S C O M *****
BASIC - COMPILER

FOR THE VZ-200 AND VZ-300
BY  BOB KITCH

(C) BY  R. B. KITCH MAR. 1987

CHANGE PARAMETERS (Y/N)?
```

If you only enter "N", you accept the existing parameters. It immediately jumps to BASIC and you can begin entering or loading your BASIC SOURCE program.

If you answer "Y", then you will be asked for each parameter one at a time.

```
STRING A. LENGTH 00032?
```

= Length of a field in a string array (0...255)

```
R. STRING LENGTH 00255?
```

= Length of a regular string variable (0...255)

With these two parameters you can greatly influence the size of the machine code program. The compiler reserves for every string variable, whether simple or in an array, the length entered above. If you have a problem with the size of the variable table, then change these two parameters.

```
VARIABLES MAXIMUM 63487?
```

= End address of the variable table

The value above is appropriate to a memory expansion with a minimum of F7FF(HEX). Only when this value is entered can the OBJECT code be test run. Always enter the optimum value for your memory expansion.

```
INPUT LENGTH   00255?
```

= Maximum length of input with the INPUT command (0...255)

You can also reduce the program size here somewhat, if you enter a smaller value than 255. Internally, a buffer of this size is set aside.

```
WARM/COLD START  COLD?
```

Here you choose, whether you want the compiled program to have a warm or cold start (see description of CHAIN).

After answering this last question, you will be returned to the introductory screen.

Entering the SOURCE program

Your SOURCE program, in BASIC, can be entered and edited via the keyboard or read in from either diskette or cassette like any other BASIC program. Before every compiling you should save (SAVE/CSAVE) the SOURCE program on diskette or cassette, so that you don't lose it due to abnormal operation of the compiler, caused by a wrong ending or an execution error during a test run.

Compiling

Call the compiler with the command.

COMPILE

It reports with,

COMPILED WITH VZBASCOM

BY BOB KITCH

and begins immediately the translation of your program.

In the course of the translation, messages will appear.

PASS1,PASS2,PASS3

to report each different pass of the compiler.

ML	BASIC	VARIABLES
nnnnn	nnnnn	nnnnn

This line gives information about the program size.

ML = Size of the created machine code
BASIC = size of the SOURCE code
VARIABLES = Size of the variable table

Don't be surprised, if the machine code program, particularly with a small BASIC program, is larger than the BASIC SOURCE program. With every translation, there is an overhead of about 1500 bytes, made up of general routines, placed at the start of the program, whether they are required or not. But this has no influence on the execution speed.

During the translation process, if a syntax error is located, it will break off with an error message. (see chapter on "Error Messages").

The program is not checked for execution errors, by the compiler (OUT OF SPACE, DIVISION BY 0 etc.). So the programmer has to avoid them through careful programmming or a test run will result in a "CRASH".

After running the compiler you will be asked for the next function.

(R)UN (B)ASIC (S)AVE

You select the desired function by entering the initial letter.

With RUN you can do a test run. This option requires the appropriate memory expansion (mimimum F7FFH).

With the entry "B", you are returned to BASIC, and you can work on your source program again.

The entry "S" allows you to save the machine code program onto diskette or cassette.

Attention

Should the SAVE function be called, with a memory expansion of less than F7FF (HEX), then the machine code program must be shifted in memory. This could wipe part of the compiler and result in it not being able to be called again. Should you want to compile a further program, you must switch off and switch on the computer and then reload VZBASCOM (Therefore: Always save the SOURCE program before compiling).

The CHAIN Option

This option allows the programmer to start additional programs one after another, without also erasing the value of variables.

CHAIN can only be used with a memory expansion having a minimum of F7FFH and works only with the cassette recorder, also the second and each further program must be started with CRUN or CLOAD.

To initialise the CHAIN option, change the parameter "WARM/COLD START", at the start of the compiler to "WARM". Now at each call the compiler will first ask,

CHAIN?

Enter "N" for the first program and "Y" for each further program.

In the second and all further programs you will be able to use previously defined and allocated variables and their actual contents.

The VZBASCOM Commands

Lascom utilises all VZ-BASIC commands except
VERIFY, CSAVE, CONT, LIST, LLIST, NEW,
PRINT# and INPUT#

As only integers can be used, the mathematical functions,

LOG, SIN, COS, TAN, ATN, EXP

are not implemented either.

Moreover none of the commands of the Disk Operating System can be used.

But VZBASCOM doesn't only have limitations, it provides in the rest of the command set, additional functions, that aren't found in VZ-BASIC.

- % corresponds to the BASIC command 'ON'
  eg. % I GOTO 100,200,300,400
  branches, depending on the contents of the variable 'I', to one of the lines listed after the 'GOTO'.

- #U scrolls the screen up

- #D scrolls the screen down

- #L scrolls the screen left

- #R scrolls the screen right

- [Individual bytes are inserted into the machine code program. With this you can directly define a machine language routine in your source program.

  [23,24 - Two bytes with the decimal values 23 and 24 are poked into the machine code program.

  [33,I - Load the register pair HL with the value of the variable 'I'.
  (33 = 21H = Z80 Command LD HL,xxxx)

  [%21,# - Load HL with the current program address.

- % Using '%' in an expression enables you to enter values in hexidecimal form.
  eg. I = %AF assigns the value 175 to I.

- MID$ Can now be used on the left side of a statement as well.

- $ With '$', a time delay in milliseconds can be programmed.
  eg. $1000 delays 1000 milliseconds (= 1 second)

- POKEW / PEEKW It is now also possible to write or read, from memory, a two byte value.
  POKEW <address>,<value between 0 and 65535>

- INPUT@ The command INPUT@ will install the same syntax and function applying to the PRINT@ command.

- PRINT@ In the PRINT@ instruction you can put more @'s in a statement.
  eg. PRINT@10,"TEST";@100,"TEST"

Exceptions to VZ-BASIC

- The command COLOR X,Y must always entered with both parameters. ie. COLOR ,X or COLOR X are not allowed.

- CLOAD and CRUN accept actual filenames, but ignore these when reading from cassette. They will always load the next program found on the cassette.

- CLOAD functions like CRUN, that is, it will always carry out an autostart.

- Variables, at the start, must not always have the value zero. So the requirement is, to be on the safe side, initialise first.

- The DIM instruction can only define one dimensional arrays. The value is entered directly as a number not a variable.
  eg. DIM A(10,10) or DIM A(X) are incorrect.
  DIM A(10) is correct.

- All arrays must be defined before they are first called. There is no standard definition as with normal BASIC.

- 'ELSE' can't be used in the last line of a program.

- A program must always be brought to a conclusion with 'STOP' or 'END'.

- In conditional statements, no string processing functions are allowed.
  eg. IF INKEY$="" THEN.... is not allowed
  nor is IF LEFT$(A$,1)="J" THEN.... instead
  use Z$=LEFT$(A$,1):IF Z$="J" THEN....
  Direct string comparison is allowed.
  eg. IF A$ = "XYZ" THEN ....

- Interlocking string processing functions are not allowed.
  eg. X$=MID$(RIGHT$(A$,5),2,2) is not allowed, instead use Z$=RIGHT$(A$,5): X$=MID$(Z$,2,2)

- An INPUT instruction can only read one string at a time.
  eg. instead of INPUT A$,B$
  use INPUT A$: INPUT B$
  INPUT works like 'LINEINPUT' on other systems.

- With the NOT function, the variable must be placed in brackets.
  eg. X = NOT (Y)

- In DATA lines text variables must always be enclosed with inverted commas.
  eg. DATA "HERMANN","ANTON" and not DATA HERMANN,ANTON

- In a FOR statement a variable can't be used after STEP.
  eg. X = 5: FOR I = 1 TO 100 STEP X is incorrect
  use FOR I = 1 to 100 STEP 5

- When using text in an INPUT statement, put in a question mark, as this is not automatically added.
  eg. INPUT"TEXT ?";A$

Error Messages

Errors in syntax, during the translation process, cause VZBASCOM to break off the translation and emit an error message.

SYNTAX ERROR IN nnnnn

This message appears as soon as VZBASCOM strikes an unfamiliar command word or when it does not find the necessary parameter.

LINE # ERROR IN nnnnn

It has tried to jump to a line which does not exist.

FOR/NEXT ERROR IN nnnnn

Either a variable has been used with STEP or there is not a FOR to go with each NEXT.

ILLEGAL STMT ERROR IN nnnnn

The compiler has found a command word that is not implemented.

VAR ERROR IN nnnnn

This is a problem with the dimensioning of an array. Possibly you forget to dimension it.

VAR OVERFLOW ERROR IN nnnnn

Overrunning the variable table. Use a smaller variable or change the standard length.

OUT OF MEMORY ERROR IN nnnnn

VZBASCOM has found that there is not sufficient room available for the machine code. The only help here is to shorten your program, or use CHAIN.

VAR/CODE CONFLICT

This is a warning message, which says, that during program run part of the OBJECT code will overwrite the Variable Table.

* PROGRAM END *

This message is displayed, if after the execution of the machine code program, it is found that the program doesn't conclude with 'STOP' or 'END'.

Some Tips

- You should be certain, before saving to disk, that the filename does not already exist on the diskette, and that the diskette is not write protected.

- Should an error occur during saving, you must immediately enter, in direct mode, the following, to restore the BASIC Pointer:

  POKE 30884,233: POKE 30885,122

  After that, use neither SAVE nor CSAVE, until the computer has been initialised using NEW.

- You should avoid loading VZBASCOM twice in a row, without switching off the computer in between. Otherwise no input is possible.

- The BASIC program memory for the SOURCE program is restricted to 9K, depending on memory expansion. Therefore, it is sensible to keep the SOURCE program as small as possible.
  - Remove all REM lines.
  - Pack as many commands as possible in a line.
  - Remove all spaces
  - Use small line numbers